Büchi Good-for-Games Automata Are Efficiently Recognizable

Marc Bagnol, Denis Kuperberg

Büchi Good-for-Games Automata Are Efficiently Recognizable

Marc Bagnol[1]
LIP, École Normale Supérieure, Lyon, France

Denis Kuperberg
CNRS, LIP, École Normale Supérieure, Lyon, France

Abstract

Good-for-Games (GFG) automata offer a compromise between deterministic and nondeterministic automata. They can resolve nondeterministic choices in a step-by-step fashion, without needing any information about the remaining suffix of the word. These automata can be used to solve games with ω-regular conditions, and in particular were introduced as a tool to solve Church's synthesis problem. We focus here on the problem of recognizing Büchi GFG automata, that we call *Büchi GFGness problem*: given a nondeterministic Büchi automaton, is it GFG? We show that this problem can be decided in P, and more precisely in $O(n^4m^2|\Sigma|^2)$, where n is the number of states, m the number of transitions and $|\Sigma|$ is the size of the alphabet. We conjecture that a very similar algorithm solves the problem in polynomial time for any fixed parity acceptance condition.

2012 ACM Subject Classification F.1.1 Models of computations, F.4.3 Formal Languages

Keywords and phrases Büchi, automata, games, polynomial time, nondeterminism

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.16

1 Introduction

The fundamental difference between determinism and nondeterminism is one of the deep questions asked by theoretical computer science. The P versus NP problem is an emblematic example of the fact that many basic questions about the power of nondeterminism are still not well-understood. In this work, we investigate an automaton model that offers a middle ground between determinism and nondeterminism, while retaining some advantages of both paradigms in the framework of automata theory. Although this model was introduced as a tool to solve a specific problem - Church's synthesis - we believe that it is a natural stepping stone on our way to get a better understanding of the power of nondeterminism in automata theory.

We will start by mentioning the historical motivation for the model of Good-for-Games (shortly GFG) automata. One of the classical problems of automata theory is synthesis – given a specification, decide if there exists a system that fulfils it and if there is, automatically construct one. The problem was posed by Church [5] and solved positively by Büchi and Landweber [3] for the case of ω-regular specifications. Henzinger and Piterman [10] have proposed the model of GFG automata, that can be seen as a weakening of determinism

[1] This work has been funded by the European Research Council (ERC) under the European Union's Horizon 2020 programme (CoVeCe, grant agreement No 678157).

while still preserving soundness and completeness when solving the synthesis problem. An automaton is GFG if there exists a strategy that resolves the nondeterministic choices, by taking into account only the prefix of the input ω-word read so far. The strategy is supposed to construct an accepting run of the automaton whenever an ω-word from the language is given. The notion of GFG automata was independently discovered in [6] under the name history-determinism, in the more general framework of regular cost functions. It turns out that deterministic cost automata have strictly smaller expressive power than nondeterministic ones and therefore history-determinism is used whenever a sequential model is needed.

We emphasize the fact that although the model of GFG automata requires the existence of a strategy resolving the nondeterminism, this strategy is not used in algorithms but only in proofs. Therefore, it is not a part of the size of the input in computations based on GFG automata. The model of GFG automata offers a compromise between determinism and nondeterminism: in particular, as deterministic automata, it preverses soundness of composition with games and trees [10, 1], while as nondeterministic automata, it can exhibit exponential succinctness compared to deterministic automata [15]. Properties of GFG automata are currently being actively investigated, and most of what we know about them has been uncovered only very recently, with several important questions still open. A brief history of recent advances in the understanding of GFG automata is given in Section 1.1 "Related works".

A major challenge in the understanding of GFG automata is to be able to decide efficiently whether an input automaton is GFG. If $\mathcal{C}$ is an accepting condition, for instance $\mathcal{C} \in \{\text{Büchi, coBüchi, Parity}\}$, we call $\mathcal{C}$ *GFGness problem* the following decision problem:

Input: A nondeterministic automaton $\mathcal{A}$ with accepting condition $\mathcal{C}$ **Output**: Is $\mathcal{A}$ a GFG automaton ?

This problem has been posed in [10], where an EXPTIME algorithm is given for the general case of parity automata. The algorithm makes use of a deterministic automaton for $\mathbf{L}(\mathcal{A})$, which can be built in exponential time. The problem is further studied in [15], where the following results are obtained:

- The coBüchi GFGness problem is in P.
- The Büchi GFGness problem is in NP.
- In general, the $\mathcal{C}$ GFGness problem is at least as hard as solving games with winning condition $\mathcal{C}$. This is tight for automata accepting all infinite words.

The precise complexity of the GFGness problem for Büchi and all higher parity conditions remained open. In particular, even for parity conditions using only 3 ranks, the only known upper bound is EXPTIME. In [14], an incremental algorithm to build GFG automata is given. This algorithm uses as a subroutine an algorithm deciding the GFGness problem. This gives an additional motivation to pinpoint the complexity of the GFGness problem, as it is a bottleneck of the algorithm from [14].

In this work, we tackle the Büchi case, and we show that the Büchi GFGness problem is in P. More precisely, we show that for a Büchi automaton $\mathcal{A}$ on alphabet Σ with n states and m transitions, we can decide whether $\mathcal{A}$ is GFG in $O(n^4m^2|\Sigma|^2)$. We do so by reducing the GFGness problem to a game where 3 tokens move in $\mathcal{A}$. The correctness of the reduction is showed using an intermediate construction using doubly exponentially many tokens.

1.1 Related Works

In the survey [7] two important results about GFG automata over finite words are mentioned: first that every GFG automaton over finite words contains an equivalent deterministic subautomaton, second that the GFGness problem is in P for automata on finite words. Additionally, a conjecture stating that every parity GFG automaton over ω-words contains an equivalent deterministic subautomaton is posed. In [1], examples were given of Büchi and coBüchi GFG automata which do not contain any equivalent deterministic subautomaton. Moreover, a link between GFG and tree automata was established: an automaton for a language L of ω-words is GFG if and only if its infinite tree version accepts the language of trees that have all their branches in L. Experimental evaluation of GFG automata and their applications to stochastic problems were discussed in [13]. In [15], it is shown that for co-Büchi automata (and all higher parity conditions), GFG automata can be exponentially more succinct than deterministic ones. For Büchi automata, this gap is not exponential, and only a quadratic upper bound is known. Typeness properties of GFG automata are established in [2], as well as complexities for changing between several acceptance conditions. In [14], the model of GFG automata is generalized to the notion of width of a nondeterministic automaton, GFG automata corresponding to width 1, and an incremental algorithm is given to build GFG automata from nondeterministic automata. The games with tokens we define in the present work are very similar in spirit to the k-simulation games introduced in [9]. However, our games cannot be seen directly as particular instances of k-simulation games, as the specific dynamics of the games are different.

2 Definitions

We will use Σ to denote a finite alphabet. The empty word is denoted ε. If $i \leq j$, the set $\{ i, i+1, i+2, \dots, j \}$ is denoted $[i,j]$. The cardinal of a set X is denoted $|X|$. If $u \in \Sigma^*$ is a word and $L \subseteq \Sigma^*$ is a language, the left quotient of L by u is $u^{-1}L = \{ v \in \Sigma^* \mid uv \in L \}$.

2.1 Automata

A nondeterministic automaton $\mathcal{A}$ is a tuple $(Q, \Sigma, q_0, \Delta, F)$ where Q is the set of states, Σ is a finite alphabet, $q_0 \in Q$ is the initial state, $\Delta : Q \times \Sigma \to 2^Q$ is the transition function, and $F \subseteq Q$ is the set of accepting states. We will often write $p \xrightarrow{a} q$ or $p \xrightarrow{a} q \in \Delta$ to signify that $q \in \Delta(p,a)$, *i.e.* there is a transition from p to q labelled by a. If for all (p,a) in $Q \times \Sigma$, $\Delta(p,a) \neq \emptyset$, we say that the automaton is *complete*. In the following, we will assume that all automata are complete, by adding a rejecting sink state $\perp$ if needed. If for all $(p,a) \in Q \times \Sigma$, $|\Delta(p,a)| = 1$, we say that $\mathcal{A}$ is *deterministic*. If $u = a_1 a_2 \dots$ is an infinite word of Σ^ω, a run of $\mathcal{A}$ on u is a sequence $q_0 q_1 q_2 \dots$ such that for all $i > 0$, we have $q_i \in \Delta(q_{i-1}, a_i)$. A run is said to be *Büchi accepting* if it contains infinitely many accepting states, and *coBüchi accepting* if it contains finitely many non-accepting states. Automata on infinite words will be called Büchi and coBüchi automata, to specify their acceptance condition. Finally, we define the *parity condition* on infinite runs: each state q has a rank $\mathrm{rk}(q) \in \mathbb{N}$, and an infinite run is accepting if the highest rank appearing infinitely often is even. An automaton on infinite words using this acceptance condition is a *parity automaton*. The language of an automaton $\mathcal{A}$, denoted $\mathbf{L}(\mathcal{A})$, is the set of words on which the automaton $\mathcal{A}$ has an accepting run. If p is a state of $\mathcal{A}$, the language accepted by $\mathcal{A}$ with p as initial state will be denoted $\mathbf{L}(p)$. A language is called *ω-regular* if it is recognized by a nondeterministic Büchi automaton, or

equivalently by a deterministic parity automaton. Two automata are said *equivalent* if they recognise the same language.

An automaton $\mathcal{A}$ is *Good-for-Games* (GFG) if there exists a function $\sigma : \Sigma^* \to Q$ (called *GFG strategy*) that resolves the nondeterminism of $\mathcal{A}$ depending only on the prefix of the input word read so far: over every word $u = a_1a_2a_3\dots$ (finite or infinite depending on the type of automaton considered), the sequence of states $\sigma(\varepsilon)\sigma(a_1)\sigma(a_1a_2)\sigma(a_1a_2a_3)\dots$ is a run of $\mathcal{A}$ on u, and it is accepting whenever $u \in \mathbf{L}(\mathcal{A})$. For instance every deterministic automaton is GFG. See [1] for more introductory material and examples on GFG automata.

2.2 Games

A *game* $\mathbf{G} = (V_E, V_A, v_I, E, W)$ of infinite duration between two players Eve (Player E) and Adam (Player A) consists of: a finite set of *positions* V being a disjoint union of V_E and V_A; an *initial position* $v_I \in V$; a set of *edges* $E \subseteq V \times V$; and a *winning condition* $W \subseteq V^\omega$.

A *play* is an infinite sequence of positions $v_0v_1v_2\dots \in V^\omega$ such that $v_0 = v_I$ and for all $n \in \mathbb{N}$, $(v_n, v_{n+1}) \in E$. A play $\pi \in V^\omega$ is *winning* for Eve if it belongs to W. Otherwise π is *winning* for Adam.

A *strategy* for Eve (resp. Adam) is a function $\sigma_E : V^* \times V_E \to V$ (resp. $\sigma_A : V^* \times V_A \to V$) describing which edge should be played given the history of the play $u \in V^*$ and the current position $v \in V$. A strategy has to obey the edge relation, *i.e.* there has to be an edge in E from v to $\sigma_P(u, v)$. A play π is *consistent* with a strategy σ_P of a player $P \in \{E, A\}$ if for every n such that $\pi(n) \in V_P$ we have $\pi(n+1) = \sigma_P(v_0 \dots v_{n-1}, v_n)$.

A strategy for Eve (resp. Adam) is *positional* if it does not use the history of the play, *i.e.* it is a function $V_E \to V$ (resp. $V_A \to V$).

We say that a strategy σ_P of a player P is *winning* if every play consistent with σ_P is winning for player P. In this case, we say that P *wins* the game $\mathbf{G}$.

A game is *positionally determined* if one of the players has a positional winning strategy in the game. A game is *half-positionally determined* if whenever Eve wins, she has a positional winning strategy.

A *finite-memory strategy* for Eve is a tuple $(M, m_0, \sigma_M, \mathtt{upd})$ where

- M is a finite set called the *memory*, and $m_0 \in M$ is the *initial memory state*.
- σ_M is a function $M \times V_E \to V$,
- $\mathtt{upd}$ is a function $M \times V \to M$ called the *update function*.

Such a tuple induces a strategy $\sigma_E : V^* \times V_E \to V$ for Eve in the original sense as follows. First, the function $\mathtt{upd}^* : V^* \to M$ is defined by $\mathtt{upd}^*(\varepsilon) = m_0$, and if $(\vec{u}, v) \in V^* \times V$, $\mathtt{upd}^*(\vec{u} \cdot v) = \mathtt{upd}(\mathtt{upd}^*(\vec{u}), v)$. We can now define σ_E by $\sigma_E(\vec{u} \cdot v) = \sigma_M(\mathtt{upd}^*(\vec{u}), v)$.

A game is *finite-memory determined* if one of the players has a finite-memory winning strategy.

▶ **Remark 1.** *In the rest of the paper, for readability purposes, we will define games in a slightly more informal manner. Namely we will allow sequences of moves of Eve and Adam going through implicit states in the game. Note that it is always possible to come back to the formal version defined in this section. We will also use examples of automata where the acceptance condition is defined on transitions rather than on states, for clarity purposes.*

2.2.1 Winning Conditions

A parity game is a game where W is a parity condition, *i.e.* where every position v has rank $\mathrm{rk}(v) \in \mathbb{N}$, and the winning set W consists of infinite words for which the maximal rank appearing infinitely often is even. The *degree* of a parity game is the number of ranks used in its parity condition.

We will use the following results on parity games:

▶ **Theorem 2** ([8, 12, 4]). *Parity games are positionally determined, and can be solved in* QuasiP.

▶ **Theorem 3** ([16, 11, 17]). *Parity games of fixed degree can be solved in polynomial time. In particular, parity games of degree* 3 *with* n *positions and* m *edges can be solved in* $O(n \cdot m)$.

If the winning set W is an ω-regular language of V^ω, we say that the game is *ω-regular*.

▶ **Theorem 4** ([3]). *ω-regular games are finite-memory determined.*

Solving an ω-regular game $G = (V, E)$ can be costly: the classical procedure from [3] is to build a deterministic automaton $\mathcal{D}$ recognizing W, and building a new game $G \circ \mathcal{D}$ of size $|V| \times |\mathcal{D}|$ with winning condition inherited from the acceptance condition of $\mathcal{D}$. Thus, the idea is to simplify the winning condition of the game at the expense of a blowup in the number of positions.

▶ **Remark 5.** *The original motivation for GFG automata [10] is that it is sufficient for the correctness of this algorithm to take $\mathcal{D}$ GFG instead of deterministic. This also explains the name "good-for-games" introduced in [10]. Remarkably, $\mathcal{D}$ can be used in this algorithm without any knowledge of the GFG strategy witnessing that $\mathcal{D}$ is GFG: as long as such a strategy exists, $\mathcal{D}$ can be used in place of a deterministic automaton to solve any ω-regular G with winning condition W.*

3 Game Characterization of GFG Automata

The main goal of this paper is to give an efficient decision procedure for the Büchi GFGness problem. In order to decide whether an input automaton is GFG, it is natural to replace the abstract definition from Section 2.1 with a more operational one.

3.1 The GFG Game

If $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ is a nondeterministic Büchi automaton recognizing a language L, let us define the *GFG game* $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$ on $\mathcal{A}$. The game is played on arena Q, starting from position q_0. Each round, from position p:

1. Adam chooses a letter $a \in \Sigma$,
2. Eve chooses a transition $p \xrightarrow{a} p'$,
3. the position of the game moves to p'.

The winning condition is the following: Eve wins if either the word $u = a_1 a_2 a_3 \dots$ chosen by Adam is not in $\mathbf{L}(\mathcal{A})$, or if the run $\rho = p_0 p_1 p_2 \dots$ she constructed is accepting (*i.e.* there are infinitely many i such that $p_i \in F$).

The GFG game actually corresponds to the original definition of GFG automata in [10]: an automaton $\mathcal{A}$ is GFG if and only if Eve wins $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$. It is shown in [1] that the definition we gave in Section 2.1 for GFG automata is equivalent.

3.2 Solving the GFG Game

Notice that $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$ is an ω-regular game. Therefore, by Remark 5, in order to solve it we need a GFG automaton for the language $W = \{(u, \rho) \in A^\omega \times Q^\omega \mid u \notin L \text{ or } \rho \text{ Büchi accepting}\}$. The Büchi condition can be recognized easily by a deterministic 2-state automaton, but for the $u \notin L$ part, we need a GFG automaton for the complement of L. A GFG automaton for L would also do, since we can consider the game where the roles of the players are reversed, thereby complementing the accepting condition. Thus, in order to decide whether an automaton for L is GFG, we seem to need a GFG automaton for L. In [15], this approach is actually used for the coBüchi GFGness problem: an auxiliary GFG automaton for L is computed, allowing to decide whether the original input automaton is itself GFG.

In the present work, we will circumvent this issue, and instead consider relaxations of the GFG game $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$, called *token games* that aim at mimicking the GFG game while enjoying a simpler winning condition.

4 Token Games

Suppose we have fixed a Büchi automaton $\mathcal{A} = (Q, \Sigma, q_0, \Delta, F)$ for the rest of this section. We define associated token games that will help deciding whether $\mathcal{A}$ is GFG.

4.1 First Attempt: the Game $\mathbf{G}_1$

As seen in Section 3.2, the difficulty of solving the GFG game $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$ comes from the fact that $\mathbf{L}(\mathcal{A})$ appears in the winning condition of $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$.

A natural attempt to circumvent this difficulty is to replace the condition "$u \notin \mathbf{L}(\mathcal{A})$" by "Adam cannot build an accepting run of $\mathcal{A}$ on u". This would simplify the winning condition, turning it into a boolean combination of Büchi conditions, thus making the game solvable in polynomial time by Theorem 3.

We therefore define $\mathbf{G}_1(\mathcal{A})$ as a modification of $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$, where in addition to choosing letters, Adam must additionally build a run witnessing that $u \in \mathbf{L}(\mathcal{A})$. If he fails to do so, Eve automatically wins the game. Therefore, we can view a play as Adam choosing letters, and both Eve and Adam possessing a token, and moving it in the automaton in order to build an accepting run. Here is a formal definition of $\mathbf{G}_1(\mathcal{A})$:

▶ **Definition 6** ($\mathbf{G}_1(\mathcal{A})$). *We define the game $\mathbf{G}_1(\mathcal{A})$ as follows. The game is played on arena Q^2, starting from (q_0, q_0). Each turn, from position (p, q):*

1. *Adam chooses a letter $a \in \Sigma$,*
2. *Eve chooses a transition $p \xrightarrow{a} p'$,*
3. *Adam chooses a transition $q \xrightarrow{a} q'$,*
4. *The game moves to position (p', q').*

Eve wins the game if either the run $\rho = p_0 p_1 \dots$ she chose is accepting, or the run $\lambda = q_0 q_1 \dots$ chosen by Adam is rejecting.

Notice that at each turn, Eve must choose a transition before Adam does. This is not abritrary, as the other way around would trivialize the game: if Adam chooses $q \xrightarrow{a} q'$ before Eve chooses $p \xrightarrow{a} p'$, then Eve can simply copy all choices of Adam, and will always win $\mathbf{G}_1(\mathcal{A})$ even if $\mathcal{A}$ is not GFG.

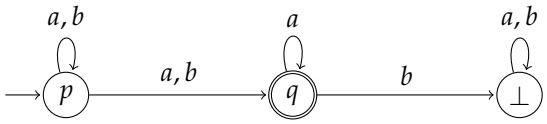
▶ **Lemma 7.** *If $\mathcal{A}$ is GFG, then Eve wins $\mathbf{G}_1(\mathcal{A})$.*

Proof. If Eve has a winning strategy in $\mathbf{G}_{\text{GFG}}(\mathcal{A})$, she can simply use the same strategy in $\mathbf{G}_1(\mathcal{A})$, ignoring the second component of the position. If the run λ built by Adam is accepting, this means the word u that has been played is in $\mathbf{L}(\mathcal{A})$, and therefore by definition of the winning condition of $\mathbf{G}_{\text{GFG}}(\mathcal{A})$, the run ρ built by Eve is accepting. ◀

The hope behind the definition of $\mathbf{G}_1(\mathcal{A})$ is that if Eve wins, then she can win without using the extra information given by the second component of the position. This would make $\mathbf{G}_1(\mathcal{A})$ equivalent to $\mathbf{G}_{\text{GFG}}(\mathcal{A})$, and allow us to decide the Büchi GFGness problem in polynomial time by Theorem 3.

Unfortunately, we can show that the converse of Lemma 7 does not hold: there is a Büchi automaton $\mathcal{B}$ such that Eve wins $\mathbf{G}_1(\mathcal{B})$ but $\mathcal{B}$ is not GFG.

Indeed, let $\mathcal{B}$ be the following automaton, recognizing $\mathbf{L}(\mathcal{B}) = (a+b)^* a^\omega$ (the accepting state is drawn with a double circle):



▶ **Lemma 8.** *The automaton $\mathcal{B}$ is not GFG, but Eve wins $\mathbf{G}_1(\mathcal{B})$.*

Proof. Adam wins $\mathbf{G}_{\text{GFG}}(\mathcal{B})$ with the following strategy: play the letter a until Eve decides to move to the state q (if she never moves, she fails to build an accepting run for a^ω which is accepted by the automaton), then play ba^ω from there; Eve is forced into state $\perp$ and cannot build an accepting run for a word of the form $a^m ba^\omega$ which is accepted by the automaton.

On the other hand, the strategy for Eve to win $\mathbf{G}_1(\mathcal{A})$ is to simply go where the token of Adam currently is. ◀

This means that in general, if $\mathcal{A}$ is a Büchi automaton, $\mathbf{G}_1(\mathcal{A})$ is not a good enough approximation of $\mathbf{G}_{\text{GFG}}(\mathcal{A})$ and does not characterize GFGness of $\mathcal{A}$.

4.2 Allowing More Tokens

Since the game $\mathbf{G}_1(\mathcal{A})$ is too easy for Eve compared to the GFG game, it is natural to try to make the game harder for Eve, by allowing Adam to build several runs in parallel, some of them being allowed to fail as long as one accepts. Indeed, it is sufficient that one accepting run exists in order to guarantee that the input word chosen by Adam is in $\mathbf{L}(\mathcal{A})$.

The game $\mathbf{G}_k(\mathcal{A})$ can be summed up as follows: Adam chooses a word, Eve moves a token in the automaton while Adam moves k tokens. After ω moves, if one of Adam's tokens followed an accepting run, then Eve's token must also have followed an accepting run. We note simply $\mathbf{G}_k$ instead of $\mathbf{G}_k(\mathcal{A})$ in the rest of the document, in order to lighten notations. Below is a formal definition of the game $\mathbf{G}_k$.

▶ **Definition 9** ($\mathbf{G}_k$). *For any integer $k \geq 2$, we define the game $\mathbf{G}_k$ as follows. The game is played on arena Q^{k+1}, starting from $(q_0, q_0, \dots, q_0)$. Each turn, from position $(p, q_1, \dots, q_k)$:*

1. *Adam chooses a letter $a \in \Sigma$,*
2. *Eve chooses a transition $p \xrightarrow{a} p'$,*

3. *Adam chooses transitions* $q_1 \xrightarrow{a} q'_1, \ldots, q_k \xrightarrow{a} q'_k$,
4. *The game moves to position* $(p', q'_1, \ldots, q'_k)$.

Eve wins the game if either the run $\rho = p_0 p_1 \ldots$ *she chose is accepting, or all runs* $\lambda_1, \ldots, \lambda_k$ *chosen by Adam are rejecting.*

▶ **Lemma 10.** $\mathbf{G}_k$ *can be seen as a parity game with* 3 *parities.*

Proof. We define a new parity condition on $\mathbf{G}_k$ as follows:

$$\mathrm{rk}(p, q_1, \ldots q_k) = \begin{cases} 2 \text{ if } p \in F \\ 1 \text{ if } p \notin F \text{ and } q_i \in F \text{ for some } i \\ 0 \text{ otherwise.} \end{cases}$$

A play is won by Eve in $\mathbf{G}_k$ iff ρ is accepting or all runs $\lambda_1, \ldots, \lambda_k$ are rejecting iff the play contains infinitely many positions with rank 2 or finitely many positions with rank 1 iff Eve wins according to the new parity condition. Notice that we use the fact that if infinitely many positions have rank 1, then there is an $i \in [1, k]$ such that the $(1+i)^{\text{th}}$ component of the position (corresponding to the i^{th} token of Adam) is in F infinitely many times. ◀

By Theorem 2, this means that $\mathbf{G}_k$ is positionally determined for all k. We will now turn to the particular case where $k = 2$, and obtain a precise upper bound on the complexity of solving $\mathbf{G}_2$. Let n be the number of states of $\mathcal{A}$ and m its number of transitions.

▶ **Lemma 11.** *An explicit version of* $\mathbf{G}_2$ *has* $O(n^3|\Sigma|)$ *positions and* $O(nm^2|\Sigma|)$ *edges.*

Proof. We can define an explicit version of $\mathbf{G}_2$, where the last component specifies which player owns the positions. Positions in this game are $V = \{(v_0, \mathrm{Adam})\} \cup \left(Q^3 \times \Sigma \times \{\mathrm{Eve}, \mathrm{Adam}\}\right)$, so $|V|$ is in $O(n^3|\Sigma|)$. Edges are

$$\begin{aligned} E = \quad & \{(v_0, \mathrm{Adam}) \to (q_0, q_0, a, \mathrm{Eve}) \mid a \in \Sigma\} \\ & \cup \{(p, q_1, q_2, a, \mathrm{Eve}) \to (p', q_1, q_2, a, \mathrm{Adam}) \mid p \xrightarrow{a} p' \in \Delta, (q_1, q_2) \in Q^2\} \\ & \cup \{(p, q_1, q_2, a, \mathrm{Adam}) \to (p, q'_1, q'_2, b, \mathrm{Eve}) \mid p \in Q, q_1 \xrightarrow{a} q'_1 \in \Delta, q_2 \xrightarrow{a} q'_2 \in \Delta, b \in \Sigma\} \end{aligned}.$$

We obtain $|E| = |\Sigma| + n^2 m + nm^2|\Sigma|$. Since we assume our automata to be complete, $n \leq m$ and $|E|$ is in $O(nm^2)|\Sigma|)$. ◀

By combining Theorem 3, Lemma 10 and Lemma 11, we obtain the following result:

▶ **Theorem 12.** G_2 *can be solved in* $O(n^4 m^2 |\Sigma|^2)$.

4.3 Some Consequences of Winning $\mathbf{G}_2$

The main result of the present work will be that Eve winning $\mathbf{G}_2$ on $\mathcal{A}$ is equivalent to $\mathcal{A}$ being GFG. One direction is actually trivial:

▶ **Proposition 13.** *If* $\mathcal{A}$ *is GFG, then Eve has a winning strategy for all* $\mathbf{G}_k$.

Proof. Eve can ignore Adam's tokens and play her GFG strategy. If the word played by Adam is in $\mathbf{L}(\mathcal{A})$ she will build an accepting run, if not Adam will not be able to build one with any of his tokens. ◀

One of the key steps of the proof is to show that if Eve wins against 2 tokens for Adam, she can actually win against any number of tokens.

▶ **Theorem 14.** *If Eve wins* $\mathbf{G}_2$ *then she wins* $\mathbf{G}_k$ *for any* k*.*

Proof. Let σ_2 be a positional winning strategy for Eve in $\mathbf{G}_2$. We proceed by induction on k, the idea being that σ_{k+1} will be obtained by having σ_k play against the first k tokens and then σ_2 against the last token and the output of σ_k. More precisely:

If Eve wins $\mathbf{G}_k$, by Theorem 2 and Lemma 10, she has a winning positional strategy $\sigma_k : Q^{k+1} \times \Sigma \to Q$ in $\mathbf{G}_k$. Define the finite-memory strategy $\sigma'_{k+1} = (M, m_0, \mu, \mathtt{upd})$ where

- the memory M is the set of states Q, and the initial memory state m_0 is q_0
- the update function is $\mathtt{upd}(m, (p, q_1, \dots, q_{k+1})) = \sigma_k(m, q_1, \dots, q_k)$
- $\mu(m, (p, q_1, \dots, q_{k+1})) = \sigma_2(p, m, q_{k+1})$ picks the move actually played by Eve

In a play using σ'_{k+1}, the memory m takes the value of σ_k playing against $q_1, \dots, q_k$ so if any of these tokens follows an accepting run, then so will m. The moves of Eve are chosen by playing σ_2 against m and q_{k+1} so that if either of these follow an accepting run, so will Eve's token. In the end, if any of Adam's tokens follows an accepting run, then either q_{k+1} or m does as well and therefore, by correctness of σ_2, the strategy σ'_{k+1} is winning for Eve in $\mathbf{G}_{k+1}$. Because $\mathbf{G}_{k+1}$ is a parity game, there exists another winning strategy σ_{k+1} for Eve that is positional. Since Eve wins $\mathbf{G}_2$, she wins $\mathbf{G}_k$ for all $k \geq 2$ by induction. ◀

Winning $\mathbf{G}_2$ also implies an important property regarding residuals, that will be key in the proof of our main theorem.

▶ **Definition 15** (residual automaton)**.** *A transition* $p \xrightarrow{a} q$ *is called* residual *if* $\mathbf{L}(q) = a^{-1}\mathbf{L}(p)$ *(remember that only* $\mathbf{L}(q) \subseteq a^{-1}\mathbf{L}(p)$ *holds in general). An automaton is* residual *if all its transitions are residual. Given an automaton* $\mathcal{A}$*, we define the associated residual automaton* $\mathcal{A}_r$ *as* $\mathcal{A}$ *where all non-residual transitions have been removed. An automaton is* pre-residual *if it accepts the same language as its residual automaton.*

▶ **Lemma 16.** *If Eve wins* $\mathbf{G}_2$ *on* $\mathcal{A}$*, we have:*

- $\mathcal{A}$ *is pre-residual,* i.e. $\mathbf{L}(A) = \mathbf{L}(\mathcal{A}_r)$
- *Eve wins* $\mathbf{G}_2$ *on* $\mathcal{A}_r$
- *If* $\mathcal{A}_r$ *is GFG, then* $\mathcal{A}$ *is GFG*

Proof. Assume that $\mathcal{A}$ is not pre-residual, *i.e.* $\mathbf{L}(\mathcal{A}) \neq \mathbf{L}(\mathcal{A}_r)$. Since $\mathcal{A}_r$ is obtained from $\mathcal{A}$ by removing transitions, we always have $\mathbf{L}(\mathcal{A}_r) \subseteq \mathbf{L}(\mathcal{A})$. So there is $u \in \mathbf{L}(A) \setminus \mathbf{L}(\mathcal{A}_r)$, *i.e.* any accepting run for u must take a non-residual transition at some point. Then Adam can win $\mathbf{G}_2$ in the following way: play the letters of u and have the first token follow Eve's token, and the second one follow an accepting run for u. If Eve never takes any non-residual transition, she cannot build an accepting run for u and loses; if she eventually takes a non-residual transition $p \xrightarrow{a} q$, then Adam picks another transition $p \xrightarrow{a} q'$ such that there is $v \in \mathbf{L}(q') \setminus \mathbf{L}(q)$, move the first token to q' and start playing the letters of v from there. Adam can build an accepting run for v from q' with the first token, while Eve is unable to do so from q. Therefore, this is a winning strategy for Adam in $\mathbf{G}_2$, a contradiction.

For the second property, we show that if Eve wins $\mathbf{G}_2(\mathcal{A})$ with a strategy σ_2, then σ_2 is actually well-defined and winning for $\mathbf{G}_2(\mathcal{A}_r)$. First note that any reachable position (p, q, r) of $\mathbf{G}_2(\mathcal{A}_r)$ has the property that $\mathbf{L}(p) = \mathbf{L}(q) = \mathbf{L}(r)$ since the initial position is (q_0, q_0, q_0) and only residual transitions can be taken. But in a position (p, q, r) such that

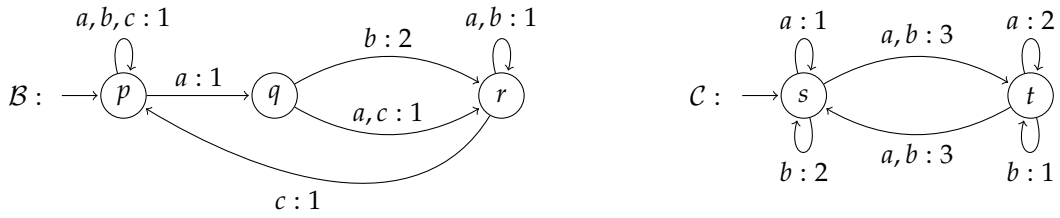
$\mathbf{L}(p) = \mathbf{L}(q) = \mathbf{L}(r)$, σ_2 cannot pick a non-residual transition $p \xrightarrow{a} p'$, otherwise Adam can start playing a word that is not in $\mathbf{L}(p')$ and win the game. So σ_2 is a valid strategy to play in $\mathbf{G}_2(\mathcal{A}_r)$. Moreover, any play of $\mathbf{G}_2(\mathcal{A}_r)$ is in particular a play of $\mathbf{G}_2(\mathcal{A})$, and we showed that $\mathbf{L}(\mathcal{A}) = \mathbf{L}(\mathcal{A}_r)$, so σ_2 is a winning strategy in $\mathbf{G}_2(\mathcal{A}_r)$.

Finally, suppose Eve has a GFG strategy σ for $\mathcal{A}_r$. Then this strategy is also well-defined on $\mathcal{A}$ and wins the GFG game because $\mathbf{L}(\mathcal{A}) = \mathbf{L}(\mathcal{A}_r)$. ◀

▶ **Remark 17.** *Any GFG automaton is pre-residual, but the converse does not hold.*

The proof that any GFG automaton is pre-residual is stated in the appendix of [15]. In our setting, it is a corollary of Proposition 13 together with the first item of Lemma 16.

We give two counter-examples for the converse: a Büchi automaton $\mathcal{B}$ on $\Sigma = \{a, b, c\}$ with $\mathbf{L}(\mathcal{B}) = (\Sigma^* ab \Sigma^* c)^\omega$, and a $\{1, 2, 3\}$-parity automaton $\mathcal{C}$ accepting $(a + b)^\omega$. In both cases, we label transitions with parity ranks.



Automata $\mathcal{B}$ and $\mathcal{C}$ are pre-residual, and in fact residual: all their states accept the language of the automaton, so all transitions are residual. However, they are not GFG: we can give a winning strategy for Adam in the GFG game in both cases.[2]

For $\mathcal{B}$, Adam first plays a. If Eve goes to q, then Adam plays abc, bringing Eve back to p. If Eve stays in p, then Adam plays bc, leaving Eve in p. Repeating this process leads Adam to build a word of $L(\mathcal{B})$, while preventing Eve from seeing any Büchi transition.

For $\mathcal{C}$, Adam can play a whenever Eve is in s and b whenever she is in t.

5 Deciding GFGness

Before we get to the sequence of results leading to the proof of our main result, let us quickly outline the approach.

We already know that if Eve is winning G_2 then she wins G_k for any k (Theorem 14) and the main idea is to find a k for which she will be able to move k tokens so that at least one follows an accepting run, and then play σ_k against these virtual tokens. We can note that by simply splitting tokens at any nondeterministic choice, she will be able to explore all the possible runs, and as k grows bigger she can keep doing it for a longer time. The results of subsection 5.1 (specifically Lemma 19) essentially guarantee there is a k large enough so that following this approach, she will eventually reach accepting states.

It then remains to use this to precisely formulate Eve's strategy to win against an hypothetic winning strategy for Adam in the GFG game, reaching a contradiction (Theorem 20).

[2] Actually, $\mathcal{B}$ has a stronger property: Eve could not win the GFG game even if she had k tokens instead of one, for any k.

5.1 Powerset Automaton

We will assume here that $\mathcal{A}$ is residual. We review a few properties of the powerset automaton that will be useful in our setting.

▶ **Definition 18.** *Given a residual Büchi automaton $\mathcal{A} = (Q, q_0, \Delta, F)$ we define the powerset automaton of $\mathcal{A}$, $2^{\mathcal{A}} = (2^Q, \Sigma, \{q_0\}, \Delta', F')$ where*

- $\Delta'(\mathbf{q}, a) = \bigcup_{p \in \mathbf{q}} \Delta(p, a)$
- $\mathbf{q} \in F'$ *when there is $q \in \mathbf{q}$ such that $q \in F$*

Note that it is well known that as such $2^{\mathcal{A}}$ does not necessarily recognize the same language as $\mathcal{A}$. However, it is always true that $\mathbf{L}(\mathcal{A}) \subseteq \mathbf{L}(2^{\mathcal{A}})$. More precisely, for any state $p \in Q$, we have $\mathbf{L}(p) \subseteq \mathbf{L}(\{p\})$. This property will be sufficient for our purpose. Let us write $\mathbf{q} \bullet w$ for the sequence of states visited by $2^{\mathcal{A}}$ when reading w from state $\mathbf{q}$.

The following lemma will be crucial in the proof of the main theorem: it tells us that if Adam is choosing letters according to a finite-memory winning strategy for the GFG game, then the number of turns before being able to see an accepting state while reading these letters is bounded. This bound will allow to follow all the possible runs up to that accepting state with a finite number of tokens.

▶ **Lemma 19.** *If τ is a finite-memory winning strategy for Adam in $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$, then there exists an integer K_τ such that: if w is a sequence of letters of length K_τ chosen by τ from a state p_0 (reached by playing τ from the starting position), and q is any state with $\mathbf{L}(p_0) = \mathbf{L}(q)$ then $\{q\} \bullet w$ contains an accepting state.*

Proof. Let M be the memory of τ and let $K_\tau = |Q \times M \times 2^Q|$. Let $w = a_1 a_2 \dots a_{K_\tau}$ be a word of length K_τ that can be played by τ in $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$ from a position p_0 reachable in $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$ with some memory m_0. Consider the sequence

$$(p_0, m_0, \{q\}) \xrightarrow{a_1} (p_1, m_1, \mathbf{q}_1) \xrightarrow{a_2} \cdots \xrightarrow{a_{K_\tau}} (p_{K_\tau}, m_{K_\tau}, \mathbf{q}_{K_\tau})$$

where the p_i's describe the states of $\mathcal{A}$ in this play, the m_i's are Adam's memory states, and the $\mathbf{q}_i$'s are the states of $2^{\mathcal{A}}$ reached upon reading the letters of w starting from $\{q\}$. By choice of K_τ, there must be $i < j$ such that $(p_i, m_i, \mathbf{q}_i) = (p_j, m_j, \mathbf{q}_j)$. This means that there is a prefix uv of w such that Eve can force the strategy τ to play uv^ω from $(p_0, m_0, \{q\})$, while guaranteeing that on the suffix v^ω, the run of $2^{\mathcal{A}}$ (corresponding to the third component) repeats the same cycle C from $\mathbf{q}_i$ to $\mathbf{q}_j = \mathbf{q}_i$.

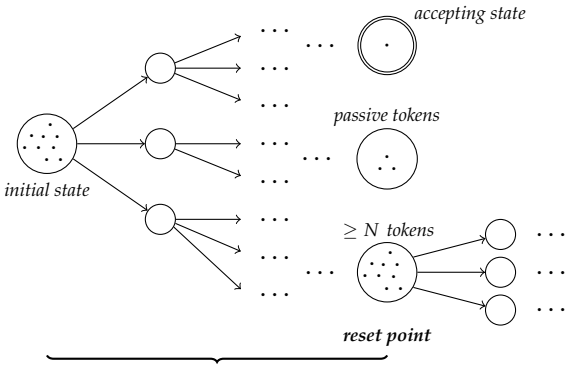
Because τ is winning for Adam, and $\mathcal{A}$ is residual, we must have $uv^\omega \in \mathbf{L}(p_0) = \mathbf{L}(q)$. Since $\mathbf{L}(q) \subseteq \mathbf{L}(\{q\})$, we have $uv^\omega \in \mathbf{L}(\{q\})$, and therefore the cycle C must contain an accepting state of $2^{\mathcal{A}}$. Since the cycle C is present in $\{q\} \bullet w$, this concludes the proof. ◀

5.2 Two Tokens Are Enough

▶ **Theorem 20.** *If Eve wins $\mathbf{G}_2$ on a residual automaton $\mathcal{A}$, then $\mathcal{A}$ is GFG.*

Proof. Assume by contradiction that Eve wins $\mathbf{G}_2$ but Adam wins $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$. By Theorem 4, he can do so with a finite-memory strategy τ (with memory of size exponential in $|Q|$). Let $K = K_\tau$ given by Lemma 19, and $c = \max\{|\Delta(p, a)| \mid p \in Q, a \in \Sigma\}$ be the degree of nondeterminism of $\mathcal{A}$. Let $N = c^K$ and $T = N \cdot |Q|$, so that when moving T tokens on $\mathcal{A}$, at least one state will hold N or more tokens at any given time. Recall that by Theorem 14,

Figure 1 Illustrating Eve's strategy for moving the T tokens



Eve has a positional winning strategy $\sigma_T : Q^{T+1} \times \Sigma \to Q$ in $\mathbf{G}_T$. Notice that T is doubly exponential in $|Q|$.

We will now define a finite-memory strategy $\sigma = (M, m_0, \sigma_M, \mathtt{upd})$ for Eve in $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$. The strategy σ will be defined according to the following intuition: Eve plays against τ by simulating T tokens moving in $\mathcal{A}$, and chooses her actual moves in $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$ by playing σ_T against these virtual tokens. The memory M of σ is Q^T, and its initial memory state is $m_0 = (q_0, \dots, q_0)$. We now describe the update function $\mathtt{upd}$ of σ. This amounts to giving a strategy for moving T tokens in $\mathcal{A}$, when letters are given by the opponent step-by-step. We will consider that some tokens are *active* and the others are *passive*. Tokens are moved according to the following rules:

- Initially, the T tokens are in q_0, and are all active.
- At each nondeterministic choice, active tokens are divided evenly between possible successors.
- Passive tokens are moved arbitrarily.
- If an accepting state is reached by some token, then choose a state p containing at least N tokens, and set the tokens in this state to active, and all others to passive. We call this a *reset point* p.

We will also consider that the initial position is a reset point. An illustration of this update strategy is given in Figure 1.

Finally, we define $\sigma_M : M \times (Q \times \Sigma) \to Q$ by $\sigma_M(\mathbf{q}, p, a) = \sigma_T(p, \mathbf{q}, a)$.

We can now consider the play ρ of σ against τ in $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$. Let $p_i, \mathbf{q}_i, a_i$ be respectively the state of Q, the memory state of σ, and the letter played by τ after i moves in ρ. Notice that since $\mathcal{A}$ is residual, for all i and $q \in \mathbf{q}_i$, we have $\mathbf{L}(p_i) = \mathbf{L}(q)$. This allows us to use Lemma 19 in the following.

We first show that there are infinitely many i such that $\mathbf{q}_i$ contains an accepting state. Consider p a reset point in the play. Starting from at least $N = c^K$ active tokens in p, and dividing them evenly at each step, the update strategy can cover all states reached by $2^{\mathcal{A}}$ from $\{p\}$ with some active tokens during K steps. By Lemma 19, the memory will reach an accepting state within these K steps, and can therefore restart at another reset point without ever running out of active tokens. This shows that there are infinitely many i such that $\mathbf{q}_i$ contains an accepting state. Since M is a finite tuple, there is one of its a components j such that the j^{th} coordinate of $\mathbf{q}_i$ is accepting for infinitely many i. By correctness of σ_T, we obtain that there are infinitely many i such that p_i is accepting. This implies that the play

ρ of $\mathbf{G}_{\mathrm{GFG}}(\mathcal{A})$ is won by Eve, a contradiction with the assumption that τ is winning for Adam. ◀

▶ **Corollary 21.** *On any Büchi automaton $\mathcal{A}$, Eve wins $\mathbf{G}_2$ if and only if $\mathcal{A}$ is GFG.*

Proof. A consequence of Lemma 16 and the above theorem: if Eve wins $\mathbf{G}_2$ on $\mathcal{A}$, then she also does on $\mathcal{A}_r$, which implies that $\mathcal{A}_r$ is GFG, and therefore $\mathcal{A}$ is GFG as well. We already saw the other direction in Proposition 13. ◀

By Theorem 12, we can now state our main result:

▶ **Theorem 22.** *The Büchi GFGness problem is in* P*, and more precisely in $O(n^4m^2|\Sigma|^2)$.*

▶ **Remark 23.** *Let us discuss briefly the possible extension of this proof to other parity cases.*

On one hand Theorem 14 is true regardless of the acceptance condition (the proof does not rely on the automaton being Büchi), which is quite promising. But on the other, the adaptation of Lemma 19 proves problematic, and without this lemma it seems difficult to find a way to move T virtual tokens so that at least one of them follows an accepting run, which we rely on critically in the proof of Theorem 20. Already in the coBüchi case a substitute technique is missing, although our current work focuses on using some of the techniques from [15] to prove Theorem 20 in this case, hoping this will eventually lead to a technique working for any parity condition.

Conclusion

We showed that the Büchi GFGness problem can be decided in P, by introducing new techniques using token games. While it seems that our proof cannot be directly used to solve efficiently the parity GFGness problem, the game $\mathbf{G}_2$ could still be relevant in this more general setting. We did not find any example of a non-GFG parity automaton $\mathcal{A}$ such that Eve wins $\mathbf{G}_2(\mathcal{A})$, so in our opinion it is plausible that Eve wins $\mathbf{G}_2(\mathcal{A})$ if and only if $\mathcal{A}$ is GFG for any parity automaton $\mathcal{A}$. Since for any fixed acceptance condition (for instance parity condition of fixed degree), the game $\mathbf{G}_2$ can be solved in P, this would put the GFGness problem in P for any fixed acceptance condition, with an algorithm that is already known.

References

1. Udi Boker, Denis Kuperberg, Orna Kupferman, and Michał Skrzypczak. Nondeterminism in the presence of a diverse or unknown future. In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part II*, pages 89–100, 2013.
2. Udi Boker, Orna Kupferman, and Michał Skrzypczak. How deterministic are good-for-games automata? In *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11-15, 2017, Kanpur, India*, pages 18:1–18:14, 2017.
3. J. Richard Buchi and Lawrence H. Landweber. Solving sequential conditions by finite-state strategies. *Transactions of the American Mathematical Society*, 138:295–311, 1969.
4. Cristian S. Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 252–263, 2017.

5 Alonzo Church. Application of recursive arithmetic to the problem of circuit synthesis. *Journal of Symbolic Logic*, 28(4):289–290, 1963.

6 Thomas Colcombet. The theory of stabilisation monoids and regular cost functions. In *Automata, languages and programming. Part II*, volume 5556 of *Lecture Notes in Comput. Sci.*, pages 139–150, Berlin, 2009. Springer.

7 Thomas Colcombet. Forms of determinism for automata (invited talk). In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, pages 1–23, 2012.

8 E. Allen Emerson and Charanjit S. Jutla. Tree automata, mu-calculus and determinacy (extended abstract). In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 368–377, 1991.

9 Kousha Etessami. A hierarchy of polynomial-time computable simulations for automata. In *CONCUR 2002 - Concurrency Theory, 13th International Conference, Brno, Czech Republic, August 20-23, 2002, Proceedings*, pages 131–144, 2002.

10 Thomas A. Henzinger and Nir Piterman. Solving games without determinization. In *Computer Science Logic, 20th International Workshop, CSL 2006, 15th Annual Conference of the EACSL, Szeged, Hungary, September 25-29, 2006, Proceedings*, pages 395–410, 2006.

11 Marcin Jurdziński. Small progress measures for solving parity games. In Horst Reichel and Sophie Tison, editors, *STACS 2000*, pages 290–301, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

12 Nils Klarlund. Progress measures, immediate determinacy, and a subset construction for tree automata. *Ann. Pure Appl. Logic*, 69(2-3):243–268, 1994.

13 Joachim Klein, David Müller, Christel Baier, and Sascha Klüppelholz. Are good-for-games automata good for probabilistic model checking? In *Language and Automata Theory and Applications - 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings*, pages 453–465, 2014.

14 Denis Kuperberg and Anirban Majumdar. Width of non-deterministic automata. In *35th Symposium on Theoretical Aspects of Computer Science, STACS 2018, February 28 to March 3, 2018, Caen, France*, pages 47:1–47:14, 2018.

15 Denis Kuperberg and Michał Skrzypczak. On determinisation of good-for-games automata. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part II*, pages 299–310, 2015.

16 Robert McNaughton. Infinite games played on finite graphs. *Annals of Pure and Applied Logic*, 65(2):149 – 184, 1993.

17 Sven Schewe. Solving parity games in big steps. *Journal of Computer and System Sciences*, 84:243 – 262, 2017.